BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Offic
Office européen des brevets

(11) Publication number: **0 621 528 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **94106131.9**

(51) Int. Cl.[5]: **G06F 3/033**, G06F 9/44

(22) Date of filing: **20.04.94**

(30) Priority: **22.04.93 US 52036**

(43) Date of publication of application:
26.10.94 Bulletin 94/43

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **MICROSOFT CORPORATION**
One Microsoft Way
Redmond, Washington 98052-6399 (US)

(72) Inventor: **Nakajima, Satoshi**
3023 168th Avenue N.E.
Bellevue, Washington 98008 (US)

(74) Representative: **Patentanwälte Grünecker, Kinkeldey, Stockmair & Partner**
Maximilianstrasse 58
D-80538 München (DE)

(54) Multiple level undo/redo mechanism.

(57) A multiple-level undo/redo mechanism is provided in an operating system and is available to application programs run on the operating system. The operating system provides a mechanism for keeping a log of user commands and providing a cursor to a position within the log. Each command may be encapsulated into an object that supports an interface for performing undo/redo operations. Similarly, the log may be encapsulated into an object that supports operations that facilitate a multiple-level undo/redo. A user may perform a single undo/redo operation, multiple successive undo/redo operations or complete undo/redo operations.

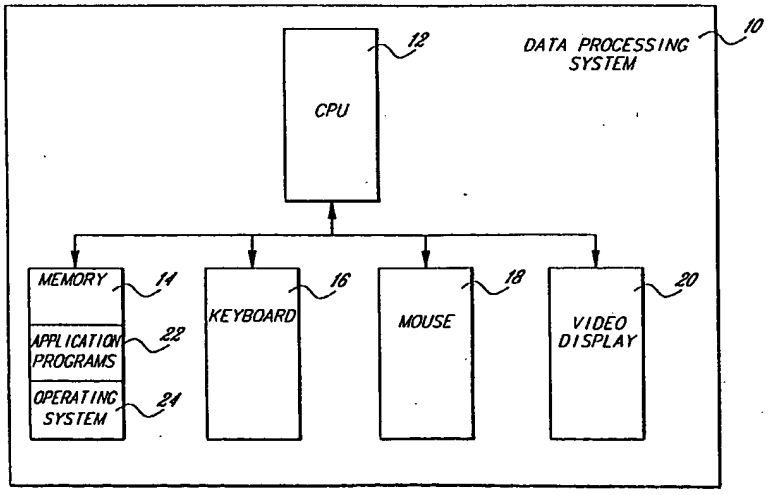

Fig. 1

EP 0 621 528 A1

Technical Field

The present invention relates generally to data processing systems and, more particularly, to a mechanism for providing a multiple level undo/redo capability in operating systems and application programs.

Background of the Invention

A single-level undo mechanism is provided by software packages, such as the Microsoft Word, version 5.0, word processing package, which is sold by Microsoft Corporation of Redmond, Washington. The single level undo mechanism allows a user command to be undone (i.e., the effects of the command are reversed) after the command has already been performed. The undo command is typically performed by selecting a menu item that lists the undo command as an option. The user is limited to a single level of undoing of commands and, thus, can only reverse the most recently executed command.

Summary of the Invention

In accordance with a first aspect of the present invention, a method is practiced in a data processing system having memory means and at least one processor that is responsive to user commands. In this method, a log of user commands that were executed by the processor is stored in the memory means. A first user command stored in a log is undone so as to reverse the effect of the first user command. Subsequently, a next user command stored in the log is undone so as to reverse the effect of the next sequential user command.

In accordance with another aspect of the present invention, a log of at least two user commands is stored in the memory means. User commands stored in the log are undone so as to reverse the effects of at least two user commands. At least two of the undone user commands are then redone so as to again execute those commands.

In accordance with a further aspect of the present invention, a list of a plurality of user commands is stored in the memory means in a sequence ranging from a selected user command that has been executed by a processor to a most recently executed user command. The effects of all of the user commands stored in a sequence of the list are undone so as to return the processor to reverse the effects of the user commands in the sequence.

In accordance with still another aspect of the present invention, a list of a plurality of user commands is stored in the memory means. The list includes user commands that have been undone. The user commands that have been undone are again executed by the processor.

Brief Description of the Drawings

Figure 1 is a block diagram of a data processing system for practicing a preferred embodiment of the present invention.

Figure 2 is a block diagram illustrating a command element object that is used in the preferred embodiment of the present invention.

Figures 3a, 3b and 3c illustrate elements of a user interface for implementing the multiple-level undo/redo mechanism of the preferred embodiment of the present invention.

Figure 4a is a flowchart illustrating the steps performed when a user enters a command in the preferred embodiment of the present invention.

Figure 4b is a diagram illustrating the state of the list of command element objects after commands are added in the preferred embodiment of the present invention.

Figure 4c is a diagram illustrating the list of command elements objects when a deletion of an element on the list is performed in the preferred embodiment of the present invention.

Figure 5a is a flowchart illustrating the steps performed in an undo operation in the preferred embodiment of the present invention.

Figure 5b is a diagram illustrating a list of command element objects when undo operations are performed in the preferred embodiment of the present invention.

Figure 6a is a flowchart illustrating the steps performed when a redo operation is performed in the preferred embodiment of the present invention.

Figure 6b is a diagram illustrating the list of command element objects when a redo operation is performed in the preferred embodiment of the present invention.

Figure 7a is a flowchart illustrating the steps performed when an undo "All the Way" operation is performed in the preferred embodiment of the present invention.

Figure 7b is a diagram illustrating the list of command element objects when an undo "All the Way" operation is performed in the preferred embodiment of the present invention.

Figure 8a is a flowchart illustrating the steps performed when a redo "All the Way" operation is performed in the preferred embodiment of the present invention.

Figure 8b is a diagram illustrating the list of command element objects when a redo "All the Way" operation is performed in the preferred embodiment of the present invention.

Detailed Description of the Invention

The preferr d embodiment of the present invention provides a mechanism for performing multiple-level undo/redo operations in an application program. The mechanism keeps a log of user commands and maintains a current position in the log to monitor a current state of the application program. The log is used by the mechanism to facilitate multiple-level undo operations and multiple-level redo operations.

Figure 1 shows a block diagram of a data processing system 10 suitable for implementing the preferred embodiment of the present invention. The data processing system 10 includes a central processing unit (CPU) 12 and a memory 14. The memory 14 holds application programs 22, an operating system 24 and other items. The data processing system 10 also includes a keyboard 16, a mouse 18 and a video display 20. The data processing system 10 shown in Figure 1 is a conventional single processor system. Nevertheless, it should be appreciated that the present invention may also be implemented in a distributed system or in other environments.

The preferred embodiment of the present invention is designed for use in an object-oriented programming environment. For purposes of the discussion below, it is assumed that the operating system 24 is an object-oriented operating system. Those skilled in the art will appreciate, however, that the present invention is not limited to use within an object-oriented operating system. Instead, it may also be implemented in other environments.

The object-oriented operating system 24 supports the use of "objects" in the data processing system 10. An object may be viewed as a combination of data members and member functions. The data members are attributes that are stored in data structures, and the member functions are functions that act upon these attributes. The notion of an object is exploited in the multiple-level undo/redo mechanism of the preferred embodiment of the present invention in that certain aspects of this mechanism are implemented as objects.

An interface is a group of semantically-related functions that are organized into a named unit. Each interface may be uniquely identified by its identifier. Interfaces have no instantiation, that is an interface definition does not include code for implementing the functions that are specified in the interface. Interfaces specify a set of signatures for functions. An object may "support" an interface. When an object supports an interface, the object provides code for the function specified by the interface. The code supplied by the object, however, must comply with the signatures specified by the interface.

The multiple-lev l undo/redo mechanism of the preferred embodiment of the present invention supports the ability for an application program to maintain a command log of user commands. The data held in the command log and functions for maintaining the data are encapsulated into a command log object. The command log object includes a list 30 (see Figure 4b) of command element objects and a cursor 32. The list 30 of command element objects is a sequential list of command element objects specifying user commands. The cursor 32 specifies a current position that corresponds with the last applied (i.e., last executed) command on the list 30 of command element objects.

Each command entered by a user during the course of execution of an application program 24 generates an associated command element object 40, like that shown in Figure 2. The command element object 40 includes a field 44 that specifies the nature of the command and a V-pointer 42. The V-pointer 42 points to a V-table 46 (i.e., having a virtual table such as found in the C++ programming language) having entries for the operations that may be performed on the command element object 40. These operations include an undo operation, a redo operation and a merge operation, for which respective entries 48a, 48b and 48c are provided. Entries 48a, 48b and 48c point to sections of code 50a, 50b and 50c for implementing their associated operations. The undo operation undoes the command at the current cursor position. The redo operation performs the next operation on the list 30 (Figure 4b). The merge operation merges command log element objects, if possible.

The command log object does not need to know about the implementation of the code 50a, 50b and 50c for implementing operations on the command element objects 40. The details of this code 50a, 50b and 50c are the concern of the command element objects 40. As such, the command log object can be implemented more easily since it does not need to concern itself with such details.

The command log object and the command element object 40 are created by the application program 22 (Figure 1). The operating system 24 provides a multi-level undo/redo facility to support multiple undo/redo operations for the application program. Part of this mechanism is a user interface. The user interface displays buttons 51 and 53 (Figure 3a) that may be activated to cause execution of undo and redo commands, respectively. The undo button 51 and the redo button 53 may be activated multiple times to perform multiple undos or redos consecutively. The user is not limited to undoing only a most recent command or redoing a most r c ntly undone command. A context m nu 55 is provided for the undo button 51 when ac-

tivated (see Figur 3b). A similar context m nu 57 (Figur 3c) is provided for the redo button. The context menus 55 (Figur 3b) and 57 (Figure 3c) provide two options: "Last Applied" and "All The Way." These options cause either only a single last applied command to be undone/redone or the commands to be undone/redone.

The four operations provided by the multiple-level undo/redo mechanism of the preferred embodiment of the present invention are perhaps best explained by way of example. As such, examples will be provided below along with the steps performed by the preferred embodiment for each of the respective four operation types.

Figure 4a is a flowchart illustrating the steps performed by the preferred embodiment of the present invention when a user enters a new command. Figure 4a will be described in conjunction with the diagram of Figure 4b. Figure 4b depicts an example of the list 30 of command element objects. Initially, a user enters a command (step 52 in Figure 4a). In the example of Figure 4b, the list 30 is initially empty. Suppose that the user then enters command "a". The multiple level undo/redo mechanism of the preferred embodiment then checks whether there are any commands on the list before the cursor 32 (step 54 in Figure 4a). If there are no commands on the list before the cursor 32, such as in the case shown in Figure 4b, the command is added to the front of the list 30 (step 58 in Figure 4a), and the cursor is moved to point to the front of the list (step 60). Thus, as shown in Figure 4b, after command "a" is entered, an entry 64 (i.e., an entry for a command element object for command "a") is added to the front of the list 30 and the cursor 32 is moved to point to entry 64. Suppose that the user now enters another command "b", (hence, repeating step 52 of Figure 4a). Steps 54, 58 and 60 of Figure 4a are then repeated so that an entry 66 (Figure 4b) is added to the front of the list 30 before the entry 64 for command "a". Cursor 32 is updated to point to entry 66 for command "b".

In the above-described fashion, the list 30 of command element objects is built. In terms of the object model described above, each time a user enters a command, an instance of a command element object 40 (Figure 2) is created; the new command element object is appended to the front of the list; and the cursor position is updated.

If in step 54 of Figure 4a it is determined that there are commands on the list 30 that are situated before the cursor 32, all the commands on the list that are before the cursor are deleted (step 56 in Figure 4a). Figure 4c shows an example of such a deletion. Suppose that initially list 30 of command log elements includes entries 68, 70 and 72 for commands "a", "b" and "c", respectively, as shown in Figur 4c. The cursor 32 points to entry 70 for command "b". Subsequently, a user enters command "d". In step 54 of Figure 4a, it is determined that entry 72 for command "c" is positioned before the cursor on list 30. Hence, entry 72 is deleted in step 56 of Figure 4a. Further, an entry 73 for command "d" is added to the front of the list 30 (see step 58 in Figure 4a), and the cursor 32 (Figure 4c) is updated to point to entry 73 (see step 60 in Figure 4a).

Once a user has built a list 30 of command element objects, such as described above, the user may execute an undo command. Figure 5a is a flowchart of the steps performed when an undo command is requested. Figure 5b is a diagram illustrating the state of the list 30 of command element objects after multiple undo commands are performed on the list. The steps of Figure 5a will be described in conjunction with the diagram of Figure 5b. Initially, a user requests an undo operation by activating the undo button 51 (Figure 3b) provided in the user interface. As was described above, a context menu 55 is displayed after the button 51 is activated and the context menu provides the user with the option of undoing only the most recent command (i.e., the "Last Applied" option). Suppose that the user selects the "Last Applied" option on the context menu 55 (step 74 in Figure 5a). The command pointed to by the cursor 32 is undone by executing code 50a (Figure 2) that is provided in the command element object (step 76) for undoing the command. In addition, the cursor 32 is decremented to point to the next successive entry on the list 30 of command element objects. To perform multiple-level undo operations, the user activates the undo button 51 multiple times to repeat the above-described steps.

Figure 5b shows an example of successive undo operations. Suppose that initially a list of command element objects includes entries 80, 82 and 84 for commands "a", "b" and "c", respectively. Further suppose that cursor 32 points to entry 84. When a user subsequently requests an undo operation, command "c" is undone and the cursor 32 is moved to point to entry 82 for command "b". If the user makes an additional undo operation request, command "b" is also undone, and the cursor is moved to point to entry 80 for command "a".

A user may also request a single redo operation. Figure 6a is a flowchart of the steps performed for a single redo operation of a most recently undone command. The process begins with a user requesting a redo operation of the most recently undone command. The user activates the redo button 53 (Figur 3c) from the user interface and then chooses the "Last Applied" option from the context menu 57 (step 86 in Figure 6a). The

command imm diately in front of the current cursor position on list 30 is th n performed (step 88), and the cursor is incremented (step 90). In terms of the object model discussed above with reference to Figure 2, the redo code 50b is executed on the command element object 40 that was most recently undone. To perform multiple-level redo operations, the user activates the redo button 53 multiple times to repeat the steps of Figure 6a.

Figure 6b shows an example of the effect of a redo command. Initially, a list 30 includes entries 92 and 94 for commands "a" and "b", respectively. Cursor 32 points to entry 92 for command "a". When the user enters a redo command, command "b" is again performed and the cursor 32 is incremented to point to entry 94 for command "b".

The user has the additional option of undoing all commands on the list 30 of command element objects. Figure 7a is a flowchart of the steps performed for a undo "All the Way" operation. Initially, a user requests that an undo "All the Way" operation be performed (Step 96 in Figure 7a). The user requests such a command by activating the undo button 51 (Figure 3b) and then selecting the "All the Way" option on the context menu 55. All user commands from the current command to the initial command are reversed (step 98 in Figure 7a). In addition, the cursor is moved to point to before the initial command on list 30 (step 100).

Figure 7b shows an example of the effect of an undo "All the Way" operation. Initially, a list 30 of command element objects 30 includes entries 102, 104 and 106 for commands "a", "b" and "c", respectively. The cursor 32 points to entry 106 for command "c". After the user has requested the undo "All the Way" operation, commands "c", "b" and "a" are sequentially undone, and the cursor 32 is decremented to point to before the first entry on the list 30.

A user may, likewise, request that the redo operation be performed "All the Way" to redo all of the commands on the list that are situated in front of the current cursor position. Figure 8a is a flowchart of the steps performed for such an operation. Initially, the user requests that the redo "All the Way" operation be performed (step 108 in Figure 8a). As with the other operations, the user selects the operation through the user interface. In particular, the user activates the redo button 53 and then selects the "All the Way" option from the context menu 57 (Figure 3c). After the selection has been made, the commands that are positioned in front of the cursor 32 on the list are performed (step 110 in Figure 8a). In addition, the cursor position is incremented to point to the front of the list (step 112).

Figure 8b shows an xample that illustrates the effect of the r do "All the Way" operation. Initially, a list 30 of command element objects includes entries 114, 116 and 118 for commands "a", "b" and "c", respectively. The cursor 32 points to entry 114 for command "a." After the redo "All the Way" operation is performed, commands "b" and "c" have been sequentially executed, and the cursor position is incremented to point to entry 118 for command "c".

While the present invention has been described with reference to a preferred embodiment thereof, those skilled in the art will, nevertheless, appreciate that various changes in form and detail may be made without departing from the present invention as defined by the appended claims.

Claims

1. In a data processing system having memory means and at least one processor that is responsive to user commands, a method comprising the steps of:
   (a) storing a log of user commands that were executed by the processor in the memory means;
   (b) undoing a first user command stored in the log so as to reverse an effect of the first user command; and
   (c) undoing a next user command stored in the log so as to reverse an effect of the next user command.

2. The method as recited in claim 1 wherein the step of storing a log of user commands further comprises the steps of:
   (i) storing each user command that was executed by the processor as an object on a list in the memory means;
   (ii) linking adjacent objects on the list; and
   (iii) storing a cursor that points to an object for one of the user commands on the list that was last executed by the processor.

3. The method as recited in claim 1 wherein the step of storing a log of user commands further comprises the step of storing the user commands sequentially in the log as the user commands are executed by the processor.

4. The method as recited in claim 1 further comprising the step of undoing all of the user commands stored in the log to reverse effects of all of the user commands stored in the log.

5. The method as recited in claim 1 further comprising th step of again executing the next command.

6. The method as recited in claim 1 further comprising the step of executing any user commands stor d in th log that have been undone.

7. The method as recited in claim 1 wherein the step of undoing the first user command further comprises the steps of:
(i) providing a user with a user interface that provides an option to the user to undo the first user command;
(ii) determining if the user selects the option to undo; and
(iii) in response to determining if the user has selected the option to undo, undoing the first user command.

8. The method as recited in claim 1 wherein the step of undoing the next user command further comprises the steps of:
(i) providing a user with a user interface that provides an option to undo the next user command;
(ii) determining if the user selects the option to undo; and
(iii) in response to determining if the user has selected the option to undo, undoing the next user command.

9. In a data processing system having memory means and at least one processor that is responsive to user commands, a method comprising the steps of:
(a) storing a log of at least two user commands in the memory means;
(b) undoing at least two of the user commands stored in the log to reverse effects of at least two undone user commands; and
(c) redoing at least two of the undone user commands so as to again execute those previously undone user commands.

10. The method of claim 9 wherein the step of storing the log of user commands further comprises the step of storing the user commands sequentially in the log as the user commands are executed.

11. The method as recited in claim 9 further comprising the step of:
undoing all of the user commands stored in the log that have not already been undone to reverse effects of those user commands.

12. The method as recited in claim 9 further comprising the step of:
executing again the user commands stored in the log that have been undone.

13. The method as recited in claim 9 wherein th step of redoing at least two of the undone user commands further comprises the steps of:
(i) providing a user with a user interface that provides an option to redo a user command;
(ii) determining if the user selects the option to redo;
(iii) in response to determining if the user has selected the option to redo, redoing a user command; and
(iv) repeating steps (i) - (iii).

14. The method as recited in claim 9 wherein the step of storing a log of at least two user commands further comprises the steps of:
(i) storing an object on a list in the memory means for each user command executed by the processor;
(ii) linking adjacent objects on the list;
(iii) storing a cursor in the memory means that points to a last executed user command on the list.

15. In a data processing system having memory means and at least one processor that is responsive to user commands, a method comprising the steps of:
(a) storing a list of a plurality of user commands in the memory means in a sequence ranging from a selected user command that has been executed by the processor to a user command that has most recently been executed by the processor; and
(b) undoing effects of all of the user commands stored on the list in the sequence so as to reverse the effects of the user commands in the sequence.

16. The method as recited in claim 15 further comprising the step of removing the undone user commands from the list of user commands stored in the memory means.

17. The method as recited in claim 15 wherein the sequence includes all of the user commands on the list and the step of undoing further comprises the step of undoing effects of all the user commands on the list.

18. In a data processing system having memory means and at least one processor that is responsive to user commands, a method comprising the steps of:
(a) storing a list of a plurality of user commands in the memory means, said list including user commands that have been undone; and

(b) x cuting again all of user commands on th list that hav been undone with the processor.

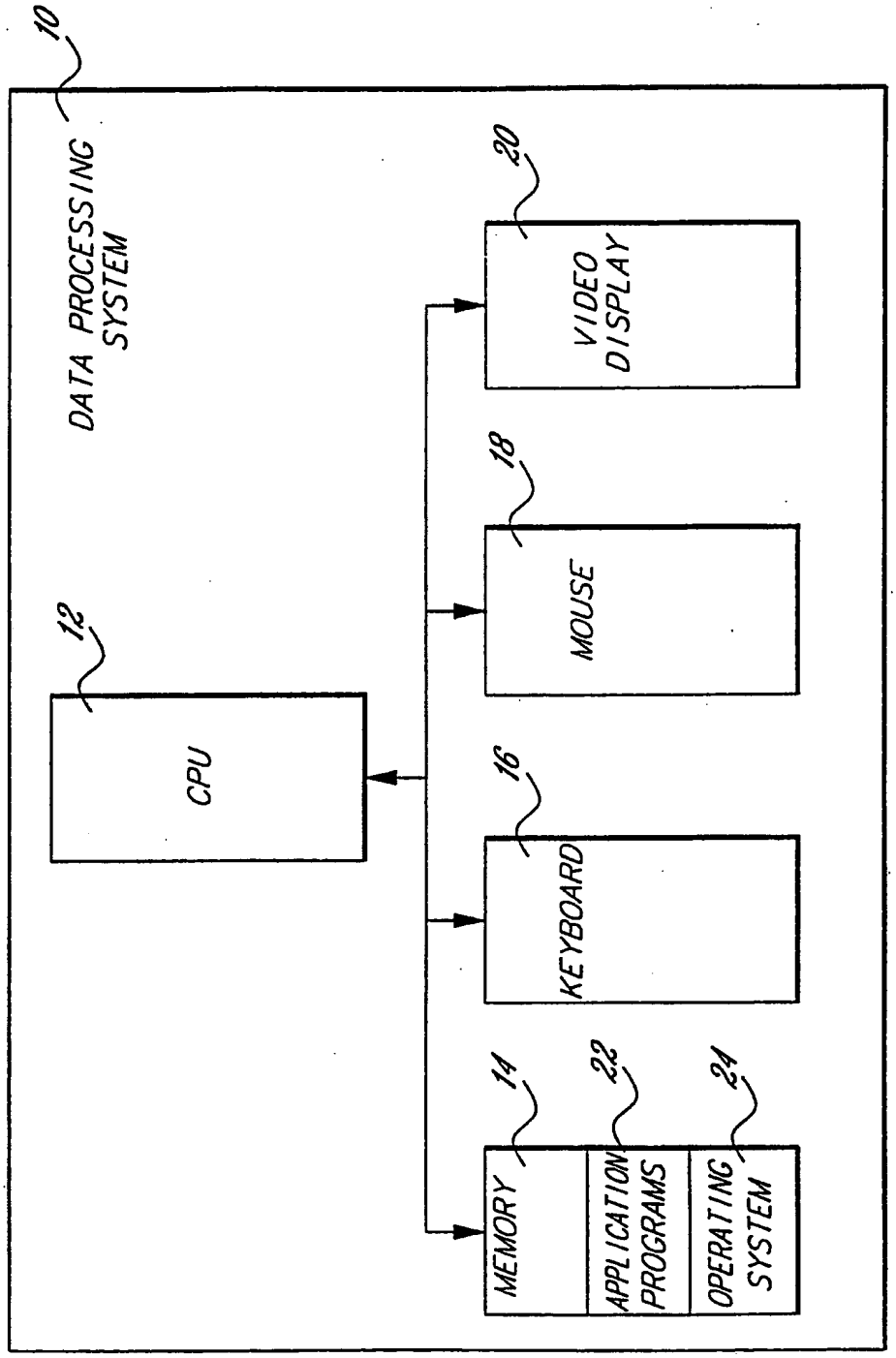
10
DATA PROCESSING SYSTEM
20
VIDEO DISPLAY
18
MOUSE
12
CPU
16
KEYBOARD
14
22
24
MEMORY
APPLICATION PROGRAMS
OPERATING SYSTEM

Fig. 1

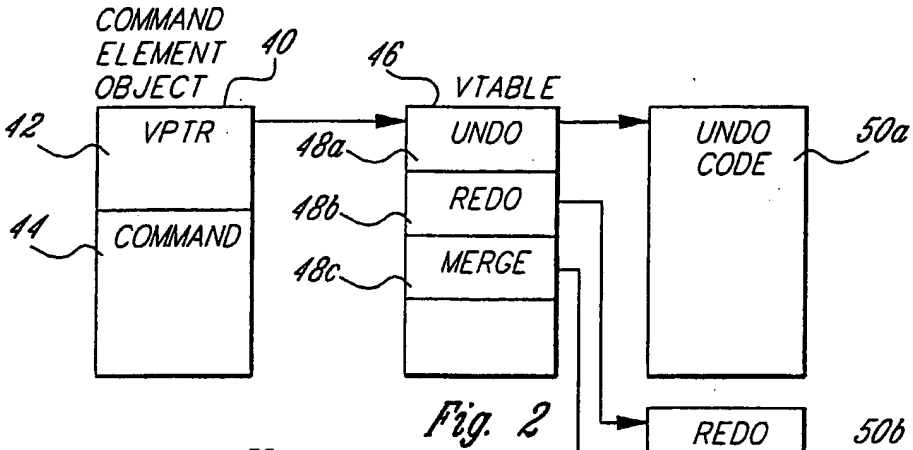

Fig. 2

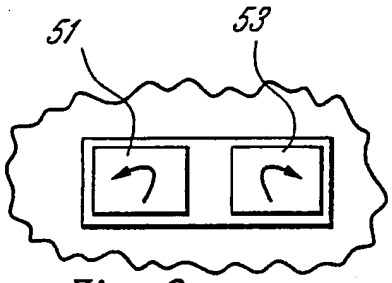

Fig. 3a

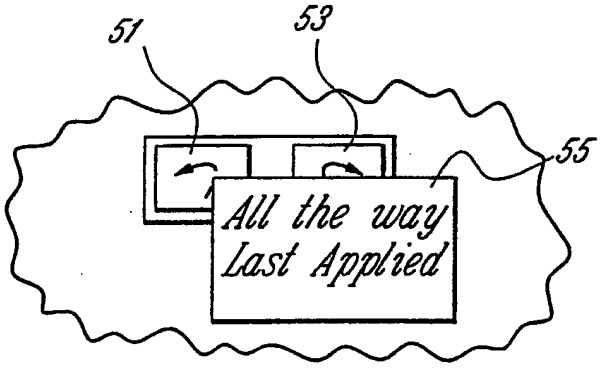

Fig. 3b

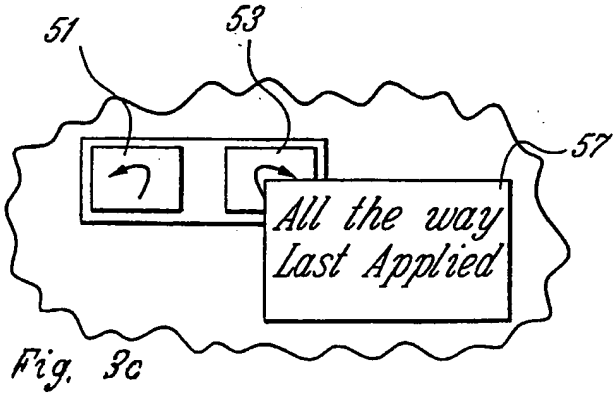

Fig. 3c

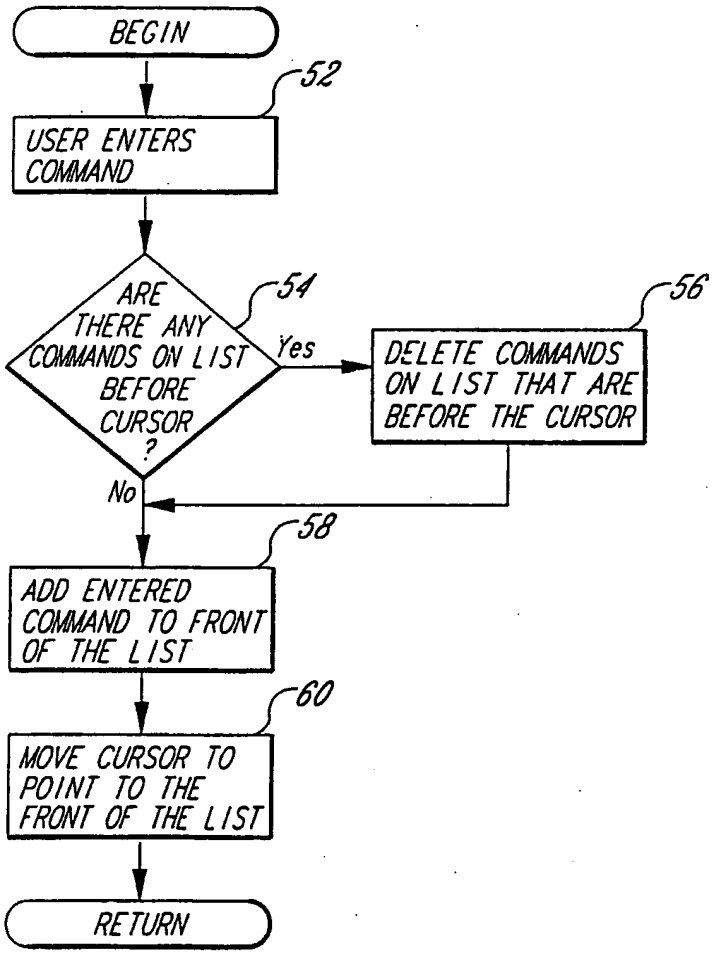
BEGIN
52
USER ENTERS COMMAND
54
ARE THERE ANY COMMANDS ON LIST BEFORE CURSOR ?
Yes
56
DELETE COMMANDS ON LIST THAT ARE BEFORE THE CURSOR
No
58
ADD ENTERED COMMAND TO FRONT OF THE LIST
60
MOVE CURSOR TO POINT TO THE FRONT OF THE LIST
RETURN

Fig. 4a

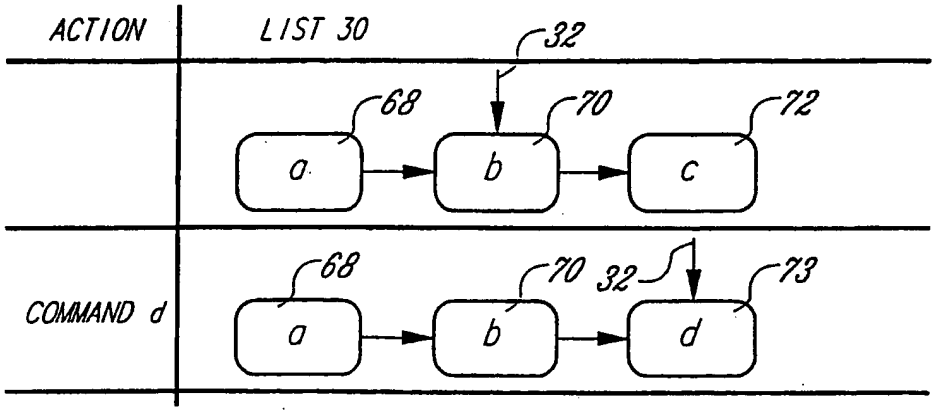
ACTION
LIST 30
32
68
70
72
a
b
c
COMMAND d
68
70
32
73
a
b
d

Fig. 4c

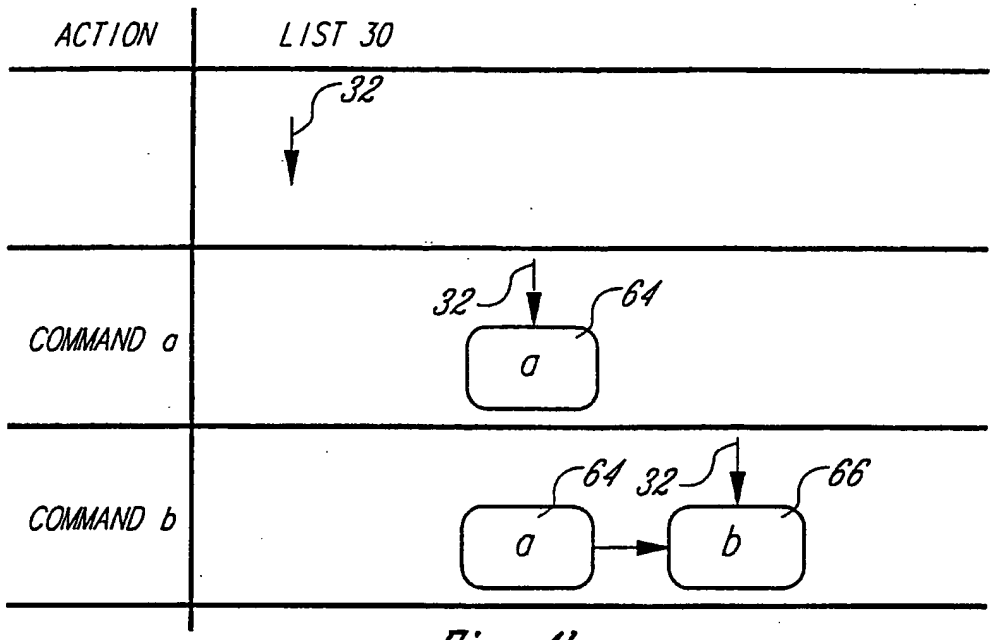
ACTION
LIST 30
32
COMMAND a
32
64
a
COMMAND b
64
32
66
a
b

Fig. 4b

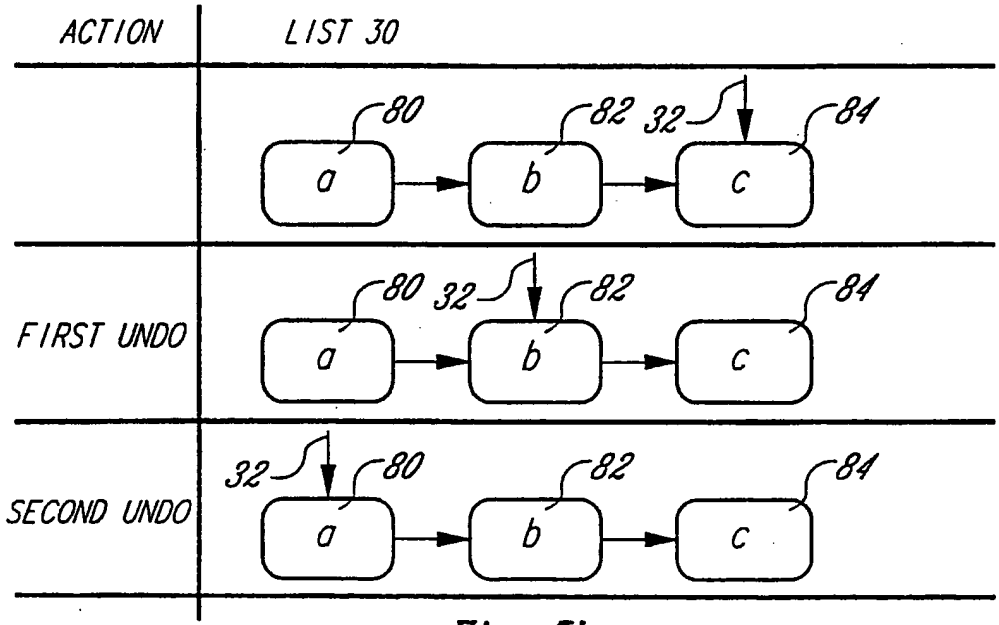
ACTION
LIST 30
80
82
32
84
a
b
c
FIRST UNDO
80
32
82
84
a
b
c
SECOND UNDO
32
80
82
84
a
b
c

Fig. 5b

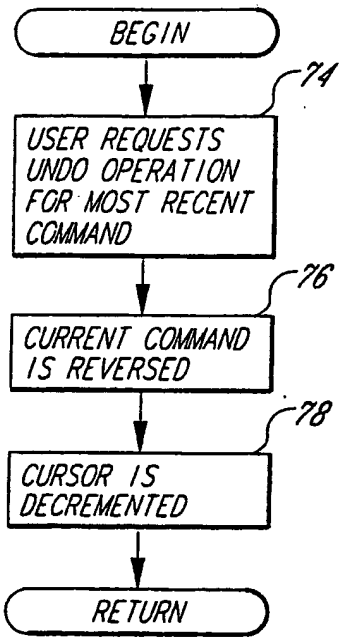

Fig. 5a

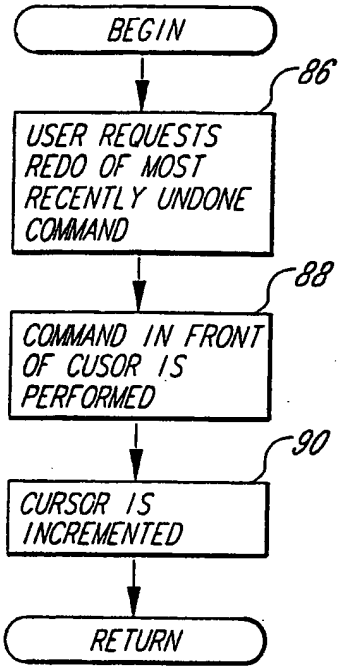

Fig. 6a

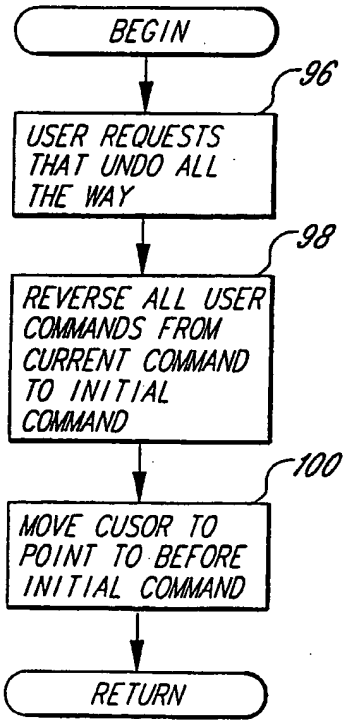

Fig. 7a

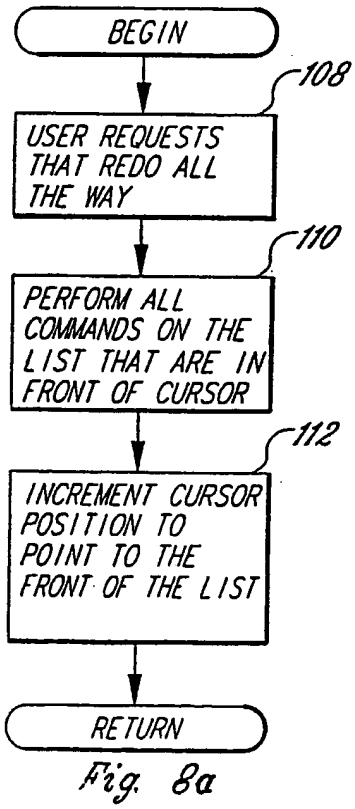

Fig. 8a

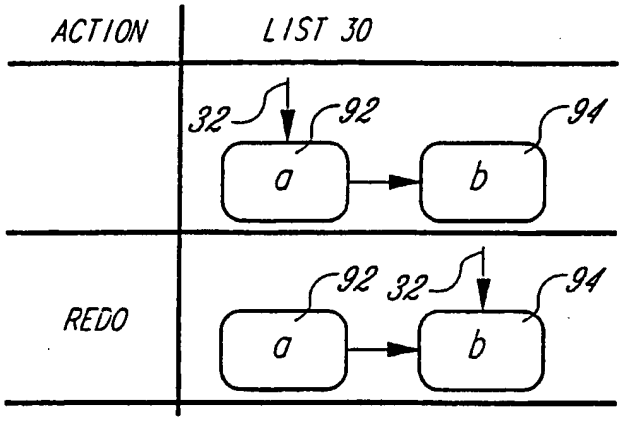

Fig. 6b

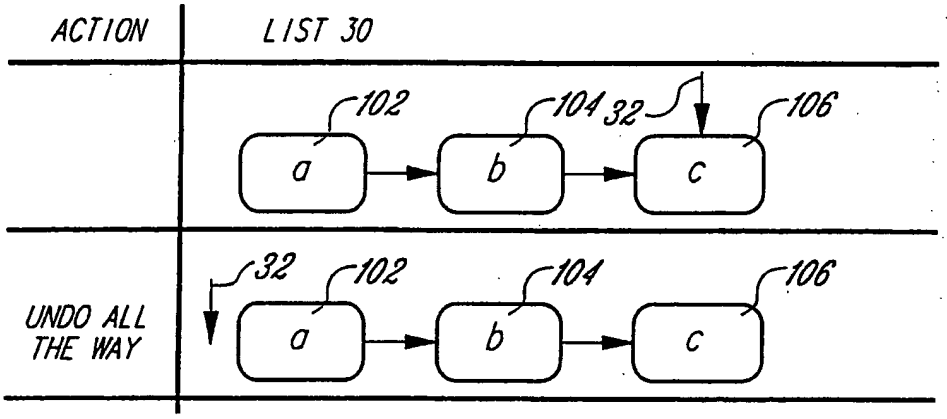

Fig. 7b

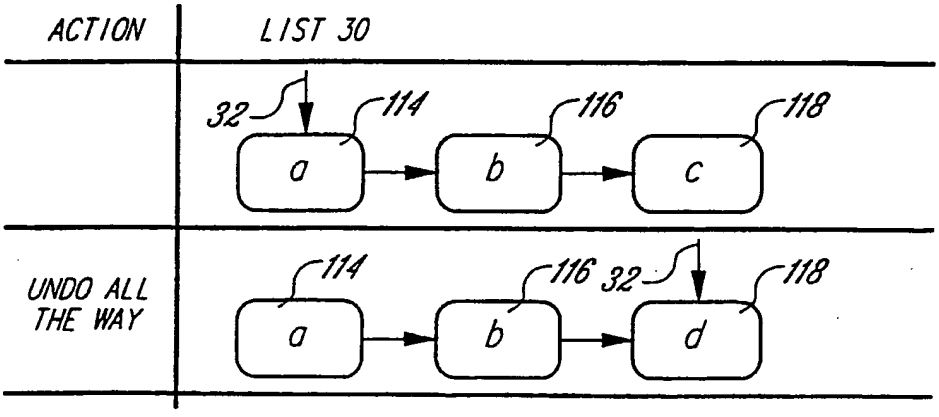

Fig. 8b

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 10 6131

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
X	IEEE SOFTWARE vol. 1, no. 4 , October 1984 , LOS ALAMITOS US pages 39 - 52 VITTER 'US&R: A new framework for redoing' * page 41, left column, line 1 - page 48, right column, line 4 * ---	1-18	G06F3/033 G06F9/44
A	4TH ANNUAL SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY 11 November 1991 , USA pages 107 - 115 WANG ET AL 'An event-object recovery model for object-oriented user interfaces.' * page 107, right column, line 3 - line 37 * * page 109, right column, line 36 - page 110, right column, line 43 * -----	1-18	

TECHNICAL FIELDS SEARCHED (Int.Cl.5)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29 July 1994	Brandt, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)